Contrasting Stack-Based and Recency-Based Back Buttons on Web Browsers

Saul Greenberg[1], Geoffrey Ho[2] and Shaun Kaasten[1]
[1]Department of Computer Science and [2]Department of Psychology
University of Calgary
Calgary, Alberta, Canada T2N 1N4
+1 403 220 6087
saul@cpsc.ucalgary.ca

ABSTRACT
People frequently use the ubiquitous Back button found in most Web browsers to return to recently visited pages. Because all commercial browsers implement Back as a stack, previously visited branches of the tree are pruned; this means that people can quickly navigate back up the tree. The problem is that previously seen pages on alternate child branches are no longer reachable through Back. An alternate method is to implement Back on a recency model, where all visited pages are placed on a recency-ordered list with duplicates removed. This means that all previously seen pages are now available via Back. Because advantages and trade-offs exist in both methods, we performed a study that contrasted how people used stack *vs* recency-based Back. We found that people have a naïve mental model of how the conventional stack-based Back works, typically perceiving it as a recency list. People are also poor predictors of what pages will be displayed with both types of Back buttons. Finally, people seem evenly split over their preference of a stack *vs.* recency-based Back button.

Keywords. History systems, revisitation, browser design.

1. INTRODUCTION

A person's ability to find and navigate effectively to new information and to new web sites is extremely important, and this has driven many researchers to understand both how people navigate within the Web, and how Web sites and browsers should be designed [e.g., 5,6]. Equally important is a person's ability to return to pages he or she has already seen: page revisitation is a regular and surprisingly strong navigational occurrence. Tauscher and Greenberg [7] found that around 60% of all pages an individual visits are to pages they have visited previously.

Given this statistic, we believe that Web browsers should go to great lengths to support effective page revisitation. Indeed, most browsers do provide revisitation support through various mechanisms: the Back and Forward buttons, history lists, bookmark facilities, and even site maps that graph the pages that a person has visited [2,7].

Of these revisitation mechanisms, it is the Back button whose use predominates: Tauscher and Greenberg [7] discovered that pressing the Back button comprised over 30% of all navigational acts. In contrast, other revisitation facilities are used infrequently e.g., < 3% for bookmarks, and < 1% for history systems. Greenberg and Cockburn [3] detailed several reasons explaining Back's popularity. First, it allows people to rapidly return to very recently visited pages, which comprises the majority of page revisits: Tauscher and Greenberg [7] found that there is a 43% chance that the next URL visited will match a member of a set containing the 10 previous visits. Second, Back requires little effort as a person merely clicks on it until the page is reached. Third, people are willing to keep Back on permanent display because it is visually compact. Finally, people can use Back successfully even when they have a naïve understanding of the way it works [1].

Back's popularity as a revisitation tool means that it deserves special attention. If we can improve it even slightly, millions will feel the added benefit. Our particular interest is in existing and alternate behaviors of the Back and Forward buttons, and how people model and use the buttons based on these behaviors. Somewhat surprising to us is the wide-spread—and unchallenged—acceptance of the stack-based navigation model underlying Back and Forward in virtually all commercial browsers.

In the remainder of this paper, we describe and contrast stack-based *vs.* recency-based Back button behavior. After summarizing these behaviors in Section 2, we introduce our study where we investigate how well people understand and use these two buttons. We present and discuss our results in Sections 4, and close by pointing out implications of our work to the design of web browsers.

2. STACK *VS.* RECENCY-BASED BACK BUTTONS

This section summarizes two different behaviors for the Back and Forward buttons: the stack-based behavior found in today's web browsers, and a recency-based behavior proposed and implemented by Greenberg and Cockburn

Greenberg, S., Ho, G. and Kaasten, S. (2000) **Contrasting Stack-Based and Recency-Based Back Buttons on Web Browsers.** *Yellow Series Report 2000-666-18*, Department of Computer Science, University of Calgary, Alberta, Canada.

http://www.cpsc.ucalgary.ca/grouplab/papers/index.html

[3]. We will illustrate these two behaviors by showing how people navigate through the small page structure shown in Figure 1.

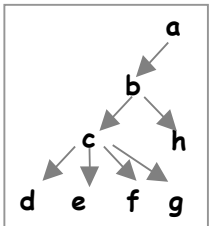


Figure 1. Example page structure.

We use the notation $x \rightarrow y$ where '$\rightarrow$' means that the person has selected or typed a link on page x to go to page y. Similarly, in $y \Leftarrow x$, the '$\Leftarrow$' means backtrack from page y to page x via the Back button. We also define *hub and spoke* navigation as an action where people follow links from one parent (the hub) to two or more children (the spokes). For example, when navigating $b \rightarrow c \Leftarrow b \rightarrow h \Leftarrow b$ in Figure 1, b acts as a hub while c and h are spokes. Of course, c could also act as a hub page if the user navigates a similar pattern to two or more of c's children. This hub and spoke behavior deserves special attention because it is a common navigational act [7] and because it results in page-pruning.

2.1 Stack

Description. The stack behavior underlying a conventional Back button has three different types of operations, as illustrated in Figure 2 and described below.

1. Clicking or typing links adds a page to the stack top.
2. Clicking Back and Forward moves the stack pointer down and up the stack respectively, displaying the page at that stack location. The actual stack contents are not altered when navigating with these buttons.
3. When the user is at any position on the stack other than the top and selects a link on a web page, all entries on the stack above the current position are popped off the stack before the new page is added. Pages popped off the stack cannot be revisited using the Back and Forward buttons.

For example, let us say a person follows the page links from pages *a* through *d* in order (Figure 1), then presses Back twice to return to page *b*, and then selects a new link on page *b* to page *h*. Figure 2a shows the stack after a person executes $a \rightarrow b \rightarrow c \rightarrow d$, where all pages were added at the stack top. In Figure 2b, the two clicks of the Back button ($d \Leftarrow c \Leftarrow b$) moves the stack pointer down the stack to *b*. Going from $b \rightarrow h$ pops pages *c* and *d* off the stack (Figure 2c), and then adds page *h* to its top (Figure 2d). Thus pages *c* and *d* are no longer reachable through the Back /Forward buttons.

Advantages and Disadvantages. The power of the stack algorithm is derived from the pruning of navigational branches that automatically occurs when users use Back followed by link selection. Essentially, each click on Back moves up one level of a tree of navigational branches and selecting a link from a position within the tree removes the lower level branches. For example, in the navigational trace described in Figures 1 and 2, the pages visited on the branches below page *b* disappeared as soon as another child of *b* (page *h)* was selected.

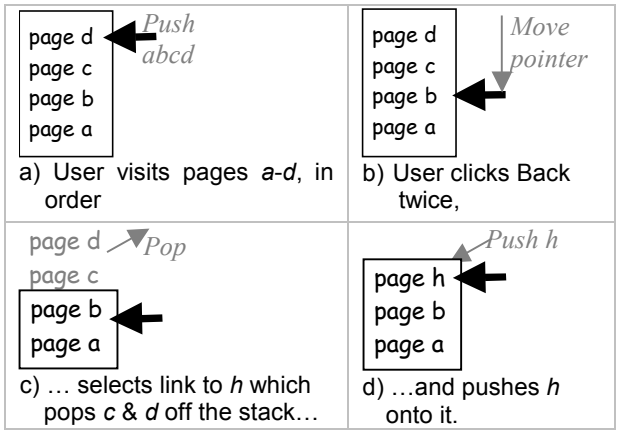


Figure 2. An example navigational trace and its effect on the stack. Note that pages *c* and *d* are popped off the stack.

This approach has some merit: after exploring a branch and selecting a new path of interest the user may no longer need the previous branch of exploration.

A counter-argument is that there are many cases where people *do* want to return to pages seen on a previously visited branch. If a person wanted to go back to page *d* from page *h* (perhaps because they now realized that information on page *d* was important), they could no longer do it via stack-based Back as page *d* has been pruned.

Still, we could argue that the Back button isn't really required for this case, because the person can first use Back to go from $h \Leftarrow b$, and then use the normal links on *b* to re-navigate $b \rightarrow c \rightarrow d$. While reasonable for short pages with few links and simple navigational paths, this could be onerous for more complex situations. Some pages are long and complex: recalling and finding the correct link within the page could be difficult for a variety of reasons. Some web pages override the coloring of previously selected links to make them indistinguishable from ones that had not been selected. Finding the correct spot to re-click on an image map may be challenging. If the person navigated a complex path to a particular page, they may find it difficult to remember and/or reconstruct that path later on.

Stack has another problem. Current systems do a poor job of communicating stack's tree-pruning behavior to its users [1] (discussed further in Section 4.1). The labels Back and Forward have affordances of linearity, rather than of a tree. There are few cues at the interface to help users distinguish between the underlying semantics of page display using link selection (popping the stack and adding to the new stack-top) versus the semantics of page display using the Back and Forward buttons (moving within the stack). Consequently, users sometimes wonder why pages are seemingly 'lost' when using Back.

2.2 Recency

Description. Perhaps the greatest disadvantage of stack-Back is that it cannot guarantee that previously visited pages are reachable by successive Back clicks. As an

alternative, we could provide a complete history of all visited pages by having Back and Forward move a person through a recency-based list, where the buttons simply navigate through the pages in reverse order to how they were seen. Surprisingly, the design of a button interface to complete history lists is not as simple as might be expected. Although forming a list of all of the pages that a user visits is trivial, designing a simple yet comprehensible interface to list traversal is complex. Greenberg and Cockburn [3] explored several models of Back based on variants of a recency-ordered history list: here we describe only the final one that they advocate: recency with duplicates removed and a temporal ordering enhancement.

We begin with a side discussion: the management of duplicate entries in the history list. Allowing duplicate pages means that the system can offer a literal representation of the order of pages that the user has seen. The disadvantage is that the list could become unnecessarily long and repetitious. Instead, we suggest pruning duplicate pages by keeping only a single copy of it in its most recent position on the list: this keeps recently revisited pages near the top. Tauscher and Greenberg [7] analyzed this, and found that substantially fewer Back presses would be required to return to a desired page when duplicates are pruned.

As with stack, navigating via links adds a page to the top of the list, and selecting Back and Forward moves a pointer through the history list (presenting the page after each click). But what happens when we select a link to a new page after using Back? We use a *temporal ordering* algorithm: Back and Forward actions move through the history list as before, but selecting a new link reorders the list to the true temporal order that matches the sequence of page as the user saw them (excluding duplicates). We implement this technique by maintaining a second pure recency list that traces the order of pages seen when a person navigates the primary history list using Back and Forward. As soon as the user displays a new page (presumably by selecting a page link), the contents of the secondary list are added in order to the main history list. Figure 3 illustrates this using the same navigation example as Figures 1 and 2. As before, visits to the pages $a \rightarrow b \rightarrow c \rightarrow d$ produces the main list $\{a,b,c,d\}$. Going from $d \Leftarrow c \Leftarrow b$ creates a second list {c,b}. As soon as the person selects the new link $b \rightarrow h$, c and then b are added to the main list giving $\{a,d,c,b,h\}$ which is the correct temporal sequence of pages (with duplicates removed) that the user has just seen. This scheme works over any number of Back and Forward actions.

Advantages and disadvantages. Recency has several potential advantages. First, it maintains the 'simple' Back and Forward button interface. Second, (and as already stated) the list of previously visited pages is complete because no pages are popped off the list. Therefore users are guaranteed to be able to revisit pages already encountered during their browsing session by using the Back button. Third, because the underlying recency list can grow indefinitely, it is feasible for Back to work *across* browsing sessions. Fourth, the temporal reordering algorithm means that users always see a temporally correct retracing of their page path when using Back.

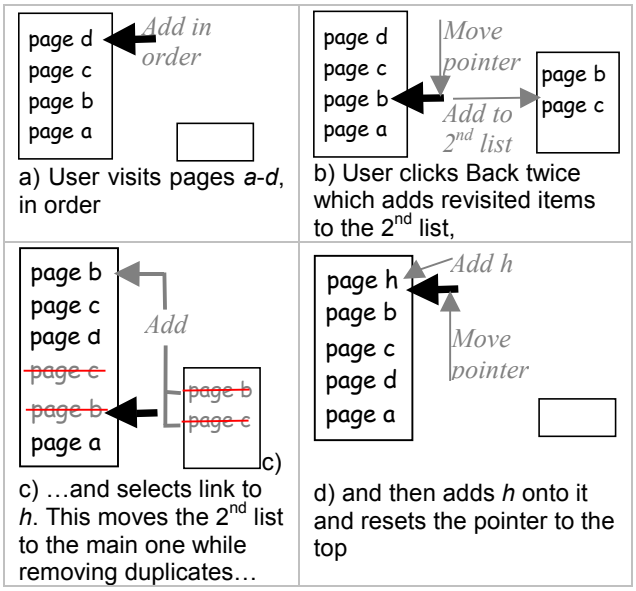


Figure 3 The same navigational trace using a recency list.

One disadvantage is if a user's goal is to navigate back up the tree (rather than to a previously seen spoke page), superfluous pages may be seen as recency does not prune the spoke pages visited on a different branch.

3. THE STUDY

Is recency a viable replacement for stack? To answer this, we designed a study that examined people's mental model of stack, and then compared how well people could predict the behavior of recency *vs.* stack-based Back navigation. We then asked people which button they preferred.

3.1 Hypotheses

H1. Users have a poor mental model of stack-Back [1].

H2. When revisiting pages over different navigational paths, users are better predictors of what pages will appear when using recency-Back *vs.* stack-Back.

H3. After using both a stack-based and a recency-based Back button for a similar set of tasks, people will prefer recency.

3.2 Subjects and expertise

Thirty volunteers participated in this study. All had some level of post-graduate education.

While subjects were not screened for Web experience, answers to a pre-test questionnaire indicated a mixed but generally web-savvy group. For web usage: 20 subjects claimed to use a browser almost every day; 5 stated their use as ranging from once every few days to once a week; while the remaining 5 reported low Web use. Subjects described themselves as: 4 being skilled experts, 13 having

good (but not expert) skills, 7 having basic skills, and the remaining 6 having beginner-level skills. All subjects stated they used Netscape Navigator and/or Microsoft Internet Explorer.

3.3 Materials

Subjects used Microsoft's Internet Explorer version 5.0 running within Windows 98 on a modern PC with a 1024x768 24-bit color display. All pages used for the study were stored on the local computer. In essence, this configuration meant that navigations and resulting page displays were uniformly rapid across pages and subjects.

Two non-standard software systems were added to the browser. The first was a Back button that used a recency with duplicates removed algorithm (Section 2.2): it was cosmetically identical to Internet Explorer's stack-based Back button. Second was an artificial search bar used for one of the tasks: while it resembled a typical search bar result, it actually contained a pre-defined static set of links.

The web sites used simulated several commercial sites. We created these sites by importing and modifying several commercial sites, chosen because they seemed typical of what we normally see on the web.

3.4 Method

Pre-test. We began with a pre-test questionnaire. In it, subjects stated their web experience as reported in Section 3.2. We then asked subjects to articulate their mental model of the conventional (stack-based) Back button they normally use (Figure 4, question 1). We assessed this understanding by giving them a simple web site—a book table of contents containing links to several chapters—and having them navigate the hub and spoke pattern: $a \rightarrow b \leftarrow a \rightarrow c$ (subjects did not see this arcane notation; they were told to go from the table of contents to Chapter 1, then Back to the contents, and then to Chapter 2). The participants were then asked how many clicks of the Back button would take them from *c* to *b* i.e., from Chapter 2 Back to Chapter 1 (Figure 4, Question 2). No matter what their answer, they were then told to try to return to *b* by actually using the stack-Back button. Afterwards, they were asked the remaining questions (3-5) about how the observed stack-based Back behavior matched the mental model previously stated in response to Question 1.

Navigation and prediction tasks. We then randomly assigned subjects to one of two groups, where each group saw either the stack- or recency-Back button first. We gave each subject five different and increasingly complex tasks. Instructions for all task are summarized below.

1. We reminded subjects to think aloud as they worked.
2. From a home page, we had subjects navigate through a series of links to a destination page. Subjects had to scan each page they saw in order to find and choose the correct next link.

3a. We asked subjects to return to a particular previously visited target page using only the Back button.

1. Describe how the back button works, and how the Back button internally manages and stores the pages you visit.

Subjects were then asked to navigate through three pages comprising a simple hub and spoke system $a \rightarrow b \leftarrow a \rightarrow c$.

2. How many times would you have to click the Back button to return to <page *b*>?

They were then told to try to return to a via Back.

3. Were there any problems?
4. Did this match your model of the back button in question 1?
5. Is the version of the Back button you are using is the same as the one supplied with your normal web browser?

Figure 4. Pre-test questionnaire excerpt concerning people's mental model of the Back Button.

3b. Before each and every Back click in step 3a, subjects had to predict what page they expected to see. They described the expected page, clicked Back, and then stated if their prediction was correct.

The tasks involve five different navigational sequences, corresponding to the sequence illustrated in Figures 5-8.

Short linear sequence (Figure 5). The subject navigates from page *a* (the home page of the Discover Alberta web site we used) through two intermediate pages *b* and *c* to reach the destination page *d* (a page describing hostels in Edmonton) i.e., $a \rightarrow b \rightarrow c \rightarrow d$; this is illustrated by the straight arrows in the Figure. The subject then uses Back to return to page *a,* making predictions before each click. Correct predictions are denoted in Figure 6 as P1 to P3 (for predictions 1 to 3). Because there are no branches, both recency and stack behave identically i.e., *a* is returned to by $d \leftarrow c \leftarrow b \leftarrow a$ (the curved arrows in the Figure).

Long linear sequence (Figure 6). This task is similar to the one above, except that more intermediary pages are involved. Reaching destination page *i* requires $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow f \rightarrow g \rightarrow h \rightarrow i$. Returning to *a* using both recency and stack Back is by $i \leftarrow h \leftarrow g \leftarrow f \leftarrow e \leftarrow d \leftarrow c \leftarrow b \leftarrow a$

Hub and spoke with return to hub (Figure 7). Reaching destination page *h* after visiting all the children of *d* requires $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \leftarrow d \rightarrow f \leftarrow d \rightarrow g \leftarrow d \rightarrow h$. Returning to revisit target hub *a* using the stack Back just goes up the hierarchy by $h \leftarrow d \leftarrow c \leftarrow b \leftarrow a$ (4 Back clicks). Recency Back takes 7 clicks, as all *d*'s children are seen again $h \leftarrow d \leftarrow g \leftarrow f \leftarrow e \leftarrow c \leftarrow b \leftarrow a$. These paths and corresponding predictions are denoted in the figure as PR1-7 for recency predictions, and PS1-4 for stack predictions.

Hub and spoke with return to spoke, then hub (Figure 7). This is almost identical to the task above (although using a different set of pages). The only difference is that subjects are first asked to revisit the child spoke page *f* of hub *d,* and then the root page *a*. Spoke page *f* is not reachable via stack as it is pruned. The revisitation path of both recency and stack Back are identical to the hub and spoke task above.

Search bar (Figure 8). This complex task simulates a user navigating through several sites by using the results of a

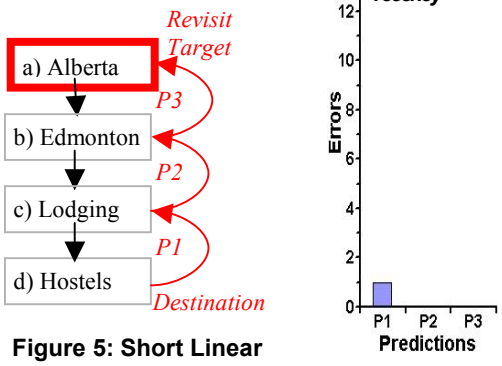


Figure 5: Short Linear

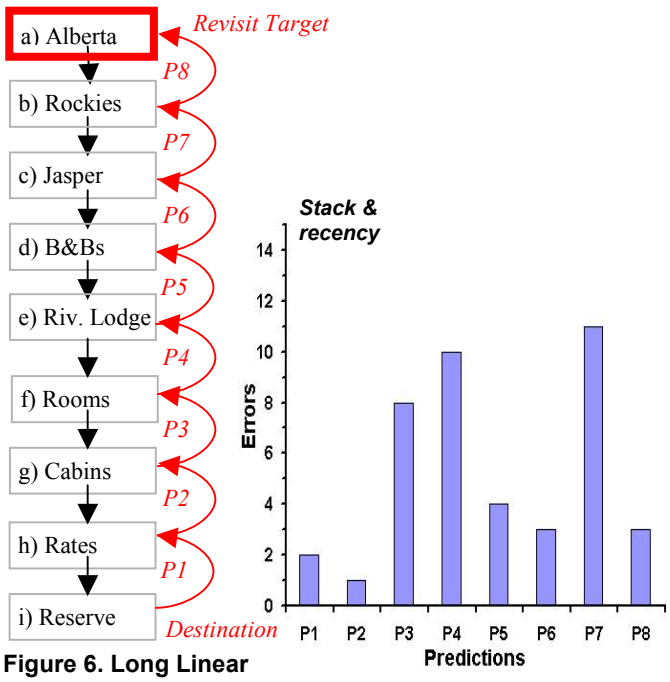
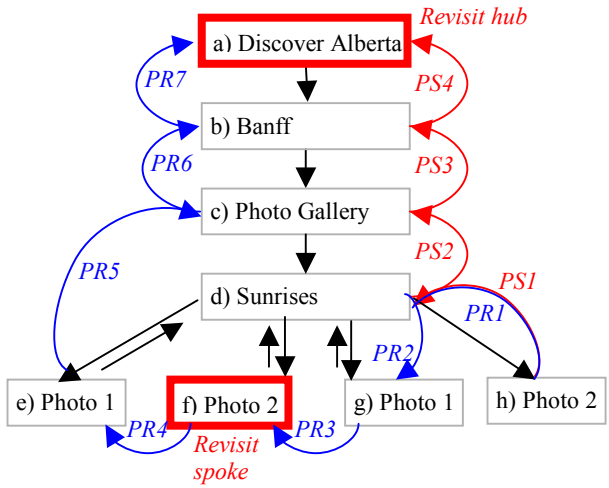


Figure 6. Long Linear



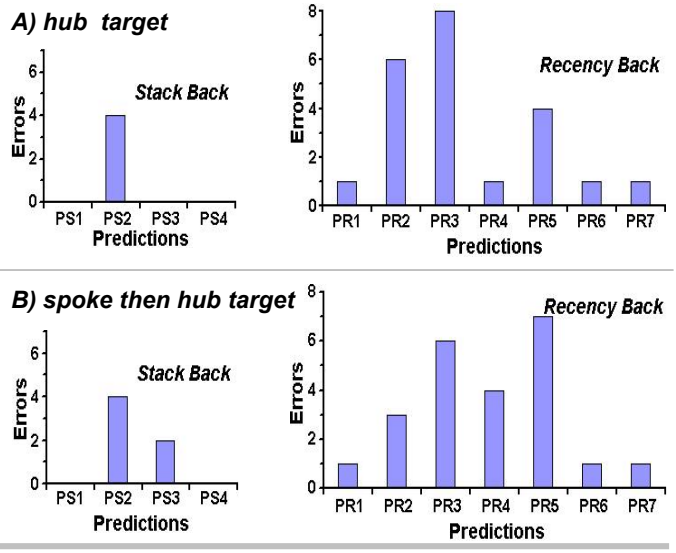
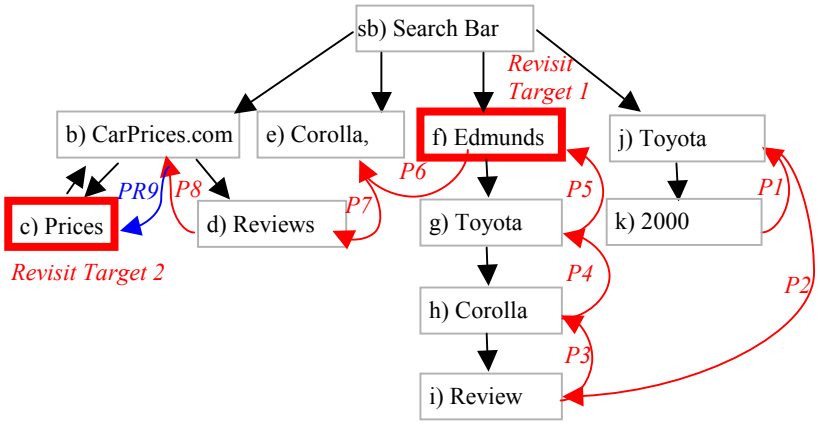


Figure 7. Hub and Spoke: a) hub and b) spoke/hub results



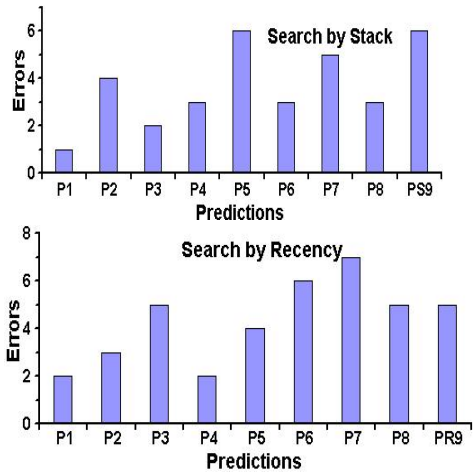


Figure 8. Search bar.

search presented in a search bar. The order of navigation by the subject (where *sb* denotes the search bar) is: *sb→b→c⇐b→d; sb→e; sb→f→g→h→i; sb→j→k.* The subject is then asked to return to target page *f*, and then to *c*. The path of stack Back is *k⇐j⇐i⇐h⇐g⇐f⇐e⇐d⇐b* (8 Back clicks). Note that because stack Back pruned spoke page *c* when the subject went back to *b*, target *c* is not reachable. Recency Back takes 9 clicks, and includes target page *c k⇐j⇐i⇐h⇐g⇐f⇐e⇐d⇐b⇐c*.

Comparison and post-test questionnaire. After completing the entire set of tasks, subjects redid them with the other type of Back button. They were then asked to comment on each Back button type and which they preferred. The entire procedure required approximately one hour.

4. RESULTS

We first describe subject's mental model of stack-Back. We then report prediction and preference data of only the first 15 subjects for reasons that will become apparent in the subsequent discussion. Afterwards, we present the preferences of the remaining 15 subjects.

4.1 Mental Model of the stack-based Back button.

When asked to describe how the conventional (stack-based) Back button worked (Figure 4 question 1), all but two had an incorrect or incomplete model of it i.e., hypothesis 1 is supported. Most simply said that it takes you back to the previously viewed pages. Some were more explicit (but still incorrect), where they said that pages are stored and displayed as a list of all pages in the order seen. A few answers hinted that subjects were aware of the stack-Back pruning behavior, but that they had an incorrect view as to when and why this happened. To quote several subjects, Back:

- goes to previous page, but sometimes you can't…I think it goes back to [a] different user;
- takes you back to previous pages in your navigation path…does seem to fail;
- takes you back to last few pages you visited but after a few clicks it takes you to the main pages (only).

Continuing the pre-test, we had people navigate a simple hub and spoke pattern, and asked them to predict how many Back clicks were required to return from the second spoke to the first spoke. The correct answer is that it is not possible, as a stack-based Back button will have pruned it off the list. However, only two users correctly answered this question (the same two that knew about the stack); the 28 others incorrectly predicted two Back button clicks.

Finally, when people were asked to try and navigate to that page via Back (only to find that they could not) most, but not all, admitted that it did not match their mental model as stated when replying to Question 1. A few said that in hindsight it did, but not for the correct reason. For example, one person said that it does match because 'there are often errors…it often doesn't work at all'. Two others said that 'some sites just do this'. Also interesting is that about half of all subjects thought that the Back button they just used did *not* behave the same as the one they normally used, even though it did.

This result accords with Cockburn and Jones' [1] study involving ten computer scientists. They too found their subject's mental model of Back was a recency-ordered list instead of a stack, and that subjects were surprised when it behaved differently than expected. As that study was done in the fairly early days of web browsers, we could have argued that today's web users are more browser-literate. This replication of their findings clearly demonstrates this is not the case: users find the exact behavior of the Back button as inscrutable now as it was then.

4.2 Predicting pages returned to by Back.

Fifteen of the 30 subjects were asked to predict what page they would see before they pressed Back as they tried to return to the revisit target. Eight of these began with recency-Back, while the remaining seven used stack-Back.

Short linear sequence. Each subject made three predictions P1, P2 and P3 for this sequence. With the exception of a single prediction error at P1 by one subject, all subjects successfully predicted what page would appear. The graph in Figure 5 illustrates this: the prediction number is on the X-axis, and the total number of errors made by the subjects is on the Y-axis.

Long linear sequence. Each subject made eight predictions P1 through P8 as they navigated back to the revisit target (Figure 6). Unlike the short linear sequence, many errors were made, as shown in Figure 6's graph. While subjects were more or less accurate for the first two predictions, errors became frequent.

Hub and spoke with return to hub. In this task, the target page was a hub up the hierarchy (the root page *a*). Four of the seven subjects using stack-Back erred only in their second prediction PS2 (Figure 7, graphs at top). That is, there seemed to be some confusion as to where Back would take them after reaching the hub page *d*. In contrast, the eight subjects using recency-Back made many prediction errors. As with stack, they were uncertain of what would happen after first reaching hub page *d* (PR2) and then again after seeing the second child (PR3)—most thought it would return to hub page *d* again. Only one person made an error on PR4, likely because they now realized they would see all children in order. However, many then expected to see hub page *d* again on PR5 rather than the hub's parent *c*.

Hub and spoke with intervening child. The only difference between this and the previous tasks is that subjects were asked to revisit the spoke page *f* before revisiting hub page *a*. We wanted to see if predictions were better or worse if they were looking for a child spoke page instead of a page up the hierarchical path. Comparing errors with the previous hub and spoke navigation with stack-based Back (where target page *f* is unreachable), we see two additional

errors on PS3 (Figure 7, graphs at bottom). We surmise that subjects thought they would see the other spoke pages at this point. Recency-Back seems to have somewhat fewer errors when going through the first few children (PS2 and PS3), although the error rate is somewhat higher afterwards. As before, many expected to see hub page *d* after each spoke was revisited.

Search bar. The error rate for predictions on this complex revisitation task was very high for both conditions (Figure 8). The only specific data point worth special mention is Prediction 9 (PS9) for the stack: all predicted another page would be seen. This is wrong: no more pages could be revisited at this point since this was the top of the stack.

Preferences. After completing all tasks with one Back button type, subjects repeated them with the other. We then asked them which one they preferred, with 9 favoring stack, 4 recency, and 2 undecided. Comments by subjects suggested that predictions were easier to make with the stack model as it skipped sub-pages.

Think-aloud. During all tasks, we observed subjects as they formed their predictions and thought-aloud about how they were making them. What was immediately obvious after running just a handful of subjects was that predicting the next page often required a great deal of cognitive work as well as time. We saw subjects try to mentally reconstruct where they had been: they would search the current page for clues as to what its parent could be while trying to recall what they had seen. They seemed to fare better on pages that had a logical hierarchical or path structure, and less so on pages whose structure was somewhat more arbitrary.

4.3 Discussion Part 1

On the surface, Hypotheses 2 and 3 are rejected: stack-Back seems better than recency. Subjects' error rate for predictions appears lower, and a large majority of subjects (9:4) preferred stack to recency.

Yet something is wrong with this story. For both conditions, we saw subjects expend a great deal of time and effort when making predictions; the error rate was also very high. This does not accord to how we see people using Back in everyday use: its selection is done without much apparent thought, and people backtrack through successive pages very quickly and successfully.

This discordance led us to rethink our study: we now believe that instructing participants to make predictions is unnatural. When users navigate, they do not usually sit and think about which page they are going to return to. Instead we suggest they use a 'click until recognize' strategy, where they simply click the button until they recognize the page being searched for. In contrast, it was clear from our observations that the participants' primary goal was predicting the subsequent page when pressing Back button; this interfered with the stated goal of returning to the appropriate revisit target page. Our prediction task forced subjects to be aware of the pages that would appear, which in turn may have led to some preference bias for stack-Back (which is likely easier to predict because one just has to reconstruct how one moves up the hierarchy).

4.4 Comparing Back and Recency without predictions.

We decided to test how participants would react to the two Back buttons if no predictions were made (hypothesis 2). Because subjects would not have to concentrate on page prediction, we expected them to stay focused on their primary goal of returning to target pages.

We continued the study with the next fifteen subjects exactly as before, except that we no longer asked them to make predictions (we omitted step 3b in the procedure outlined in Section 3.4). Subjects used both types of buttons over all tasks, and then stated their preferences.

Unlike the previous results where subjects favored stack, this new set of subjects had a slight preference for recency. Eight preferred recency, six preferred stack, and one was undecided. People who preferred recency commented:

- Go through the actual order more than not
- Pages come back sequentially as they should
- More predictable: goes through the actual order
- Doesn't feel like more clicking
- Stack missed a whole bunch of pages
- More intuitive. Liked [that it had] no duplicates

People who preferred stack commented:

- More used to it
- Recency produced extra clicks
- Doesn't take you back to sub-pages

4.5 Discussion Part 2

Unlike the first fifteen subjects that preferred stack-Back when making prediction, this set slightly preferred recency-Back. That is, hypothesis 3 is now weakly supported.

This change in attitude re-enforces our conviction that people do not have an exact model of what pages they expect to see as they use Back, and that they use a 'click until recognize' strategy instead.

One recurring comment made in regards to the recency model was the inability to recover 'hub' pages: subjects expected to see the hub page after each visit to a child. The recency with duplicates removed algorithm only showed the hub page once, but the perception of the users was that it did not. Instead, they expected a pure sequential model. Does this suggest that hub pages should be duplicated rather than only shown once? We think not. With more extended use of recency Back, users may realize that the hub pages are accessible and may find the duplication of pages unnecessary. We also believe that seeing hub pages several times will introduce a different type of confusion i.e. that Back is merely cycling through the same sequence of pages. Still, more testing is needed before drawing a final recommendation of how duplicates are handled.

5. IMPLICATIONS TO BROWSER DESIGNERS

At first impression, there is no compelling reason to change the current stack-based Back button to a recency-based one. People seem comfortable with Stack, even though they have a poor model of it. This is because their 'click until recognize' strategy does not require an accurate model of its behavior. As well, people seem unconcerned about the mysterious disappearance of pruned pages. Importantly, there is no overwhelming preference by our subjects of recency-Back over stack. While we could still argue that recency is better than stack because no pages are lost, using it as a replacement for the familiar stack-Back idiom could introduce unnecessary risk.

However, the slight preference of recency over stack means that new designs can include recency with no penalty. One possibility, which we have implemented in a prototype web browser, is to include both stack and recency-based buttons. We relabel Stack Back and Forward as Up and Down, as this more accurately reflects the semantics of moving up and down the navigational hierarchy that is a side product of pruning. Back and Forward are now recency-based, as they reflect the semantics of moving back and forward on the recency-ordered history list. Nonetheless, we recommend caution. Because users have a fuzzy notion of how stack and recency behave, the differences between these buttons may be unclear to them. As well, it adds complexity: yet another decision must be made as to which revisitation method should be chosen.

Perhaps a more compelling reason for using a recency-based Back button is to remove the differences between how Back and other revisitation systems work. In related work, we are designing a new history system that integrates Back, history, and bookmarks by unifying them to operate over a single recency-based list [4]. Back and Forward simply become mechanisms that navigate in linear order item by item through the combined history / bookmark list. If the history list is visible, then items are highlighted as the user selects Back. This visually re-enforces how Back works.

6. SUMMARY

Even though Back is probably the most highly-used interface widget in existence today, there are (to our knowledge) no other studies that scrutinize alternatives to the widely-accepted stack algorithm. In this paper, we proposed a recency-based behavior with duplicates removed. We showed that people have a poor model of both stack and recency Back, and that they make many errors when predicting what pages will appear. We suggest that this is not really a problem as people employ a 'click until recognize' strategy, where they simply click Back until they recognize the desired page. We also showed that when people had the opportunity to use both buttons, slightly more of them preferred recency to stack.

Because this preference of recency is not overwhelming, we advocate the replacement of stack-based Back with recency only if other design considerations warrant them. We feel that good design opportunities do exist, especially for a recency-based Back to be integrated with a recency-based history list to produce a single model of how pages can be revisited.

There is no question that the high usage rate of Back warrants further research: millions will be affected by even a small improvement in its design.

Acknowledgements. Microsoft and NSERC for funding our research. Special thanks to Kent Sullivan, Robert Graf, and Linda Tauscher for their intellectual contributions.

Study materials. In the interest of replication, all study materials (including the experimental recency-based back button) are available at: http://www.cpsc.ucalgary.ca/grouplab/software/.

REFERENCES

1. Cockburn, A. & Jones, S. Which way now? Analysing and easing inadequacies in WWW navigation. *Int J Human-Computer Studies* **45**(1), 105-129, 1996.
2. Cockburn, A. & Jones, S. Design issues for World Wide Web navigation visualisation tools. *Proc RIAO'97: The 5th Conference on Computer-Assisted Research of Information*. McGill University, Canada, 55-74, 1997.
3. Greenberg, S. & Cockburn, A. Getting back to Back: Alternate behaviors for a web browser's Back button. *Proc 5th Annual Human Factors and the Web Conference*, NIST, Gaithersburg, USA, 1999.
4. Kaasten, S. and Greenberg, S. Designing an integrated bookmark / history system for web browsing. *Proc. Western Computer Graphics Symposium*, March 26-29, 2000. Available from http://www.cpsc.ucalgary.ca/grouplab/papers/000/00-GrouplabPapers.Skigraph/kaasten/kaasten.pdf
5. Nielsen, J. *Designing Web usability.* New Riders, 2000.
6. Sano, D. *Designing large-scale web sites: A visual design methodology.* Wiley Computer Publishing, 1996.
7. Tauscher, L. & Greenberg, S. How people revisit web pages: Empirical findings and implications for the design of history systems. *Int J Human Computer Studies*, **47**(1), 97-138, 1997.